Algorithm which always finds all Roots of an n-th Degree Polynomial

Dr. Klaus Huber
Huber Consult
Sesenheimer Str. 21
10627 Berlin
Germany
email: filterdesign@klaus-huber.net

Abstract

An algorithm is presented which finds all roots of an n-th degree polynomial. In contrast to the widely-known algorithms of Newton or Laguerre, which in the general case of complex zeros are not guaranteed to converge, the algorithm always finds all solutions.

Index Terms — Root finding, Hurwitz test, Filter design.

1 Introduction

In this contribution an algorithm is presented which finds all roots of an n-th degree polynomial $f(z) = \sum_{j=0}^{n} f_j z^j$, where f_j are real numbers. If all roots of $f(z)$ are real, it is well-known that the algorithm of Laguerre [5] is able to find them with certainty. If all or some of the roots are complex, Laguerre's algorithm still is usually the preferred algorithm, but the convergence to the roots of $f(z)$ is not guaranteed, the convergence depends on the starting point of the iteration as is the case to a much higher degree for the method of Newton.

Lehmer [6] has already given an algorithm which always finds all roots. However, Acton ([1], p.197) criticizes Lehmer's algorithm that it *finds roots about as precisely as one of the unattractive standard packages but takes three times as long.* It is not the aim of this paper to take part in this discussion, but to present another algorithm – much simpler than Lehmer's – which with certainty finds all roots (complex or real) of the polynomial $f(z)$.

Having an algorithm which always finds the roots of polynomials is a practical necessity. For example in filterdesign (see e.g. [2] for a recent application) it is important for certain designs to solve such equations and it is rather annoying if users of a software package must interact (or worse do not get results) for designs which are realisable.

2 Basics of the Algorithm

The basic part of the new algorithm is a well-known method for determining whether all roots of a polynomial are in the left complex plane[1]. This problem has been solved

[1] Being able to answer this question rapidly without actually finding the roots has an enormous significance for engineering applications.

by Hurwitz [3]. The resulting test is usually called Hurwitz-test or Hurwitz-criterion. In its most streamlined form known to filter theorists (and for all others summarized in [4], p.293) we simply develop the ratio of the even and odd parts of the polynomial $f(z)$ into a continued fraction and look whether all coefficients are positive. Then $f(z)$ has all its roots in the left half plane.

Example 1 *Consider* $f(z) = z^4 + 5z^3 + 10z^2 + 10z + 4$. *To prove that* $f(z)$ *has all its roots in the left half plane we compute*

$$\frac{z^4 + 10z^2 + 4}{5z^3 + 10z} = \frac{1}{5}z + \frac{1}{\frac{5}{8}z + \frac{1}{\frac{16}{15}z + \frac{1}{\frac{15}{8}z}}}$$

and find that all $deg\{f(z)\} = 4$ *coefficients* $\frac{1}{5}, \frac{5}{8}, \frac{16}{15}, \frac{15}{8}$ *are positive.*

If the degree n of $f(z)$ is odd the ratio of the odd and even parts are develeped into a continued fraction. We say that $f(z)$ is a Hurwitz polynomial if it has all roots in the left half plane[2]. Hurwitz's test is easily programmed.

To determine the real part of a root (real or complex) of $f(z)$ now is very easy. We simply test whether $f(z)$ is a Hurwitz polynomial or not. Then, we simply shift the polynomial by a suitable real number a to the right or the left, i.e. we compute $f(z+a)$ or $f(z-a)$ and again make a Hurwitz test. In this way we can determine in a bisection-like iteration by interval halving the number σ which approximates the real part of a zero of $f(z)$ to any precision desired.

A suitable starting number a can easily be obtained from a well-known result of Cauchy (or refinements thereof). Here, for simplicity, we choose Cauchy's result. There are better methods which can e.g. be found in Zeheb[8] and the references therein. Often the problem one has to solve gives better initial conditions (e.g. in filter design one knows that a lowpass prototype filter has the roots in a rectangle in the left half plane). Cauchys result tells us that all roots of $f(z)$ are located within a radius around the origin which is smaller than $r_C = 1 + \max_{k=0..n-1} |\frac{f_k}{f_n}|$. Hence, given a polynomial $f(z)$ we know without any test that $f(z + r_C)$ is a Hurwitz polynomial and $f(z - r_C)$ is not a Hurwitz polynomial. A pseudo-code implementation (Algorithm 1) is given in the following figure. Indentation is used instead of *begin end* commands. For simplicity the error bound ϵ is assumed to be a global variable.

Algorithm 1: Bisection Algorithm for Determination of Real Part σ

```
bisect(f(z))
  r_C ← cauchyradius(f(z))
  (a, b) ← (−r_C, r_C)
  while |b − a| > ε :
    σ ← (a + b)/2
    g(z) ← f(z + σ)
    if hurwitz(g(z)): b ← σ
    else : a ← σ
  return (a+b)/2
```

[2]Note that a necessary but not sufficient condition for $f(z)$ being a Hurwitz polynomial is that all coeffients $f_j \neq 0$ and that all f_j have the same sign.

Clearly, the error bound ϵ must be selected according to the problem given and also if necessary (for example if the polynomial is ill-conditioned or two zeroes are very close together) one has to select multiprecision arithmetic. The selection of ϵ will be treated below.

Once we have found the real part σ of a zero of $f(z)$, the imaginary part is found as follows. We expand $f(\sigma + i\,y)$ (where $i = \sqrt{-1}$) which gives a complex polynomial which can be separated into a real polynomial $u(y)$ plus i times another real polynomial $v(y)$:

$$f(\sigma + i\,y) = \sum_{j=0}^{N} f_j \cdot (\sigma + i\,y)^j = u(y) + i \cdot v(y)\ .$$

At a zero we have $f(\sigma + iy) = 0$, i.e. we are looking for values y where both polynomials are zero, i.e. $u(y) = 0$ and $v(y) = 0$. Hence we can find y by computing the greatest common divisor (gcd) of $u(y)$ and $v(y)$. Clearly there may be more than one solution. Only for a real zero is the value $y = 0$ unique. A conjugate pair of complex zeros delivers an equation in y^2. If there are more than one or two solutions on the line $x = \sigma$ we may get a $gcd(u, v)$ polynomial in y of degree greater equal than three. The resulting equation (if it has degree higher than three or four) in turn then can also be treated with our new algorithm (or with Laguerre's method, as all roots are real). The zero (or zeros) found can then be deflated and the remaining zeros can be tackled in the same way.

To this end we prove that the greatest common divisor of $u(y)$ and $v(y)$ has only real roots if the roots of $f(z)$ are found and deflated - as is the case in our algorithm - with the rightmost zeros first.

Theorem 1 *When deflating the rightmost zero or zeros of $f(z)$ first, the greatest common divisor $p(y)$ of $u(y)$ and $v(y)$ has only real roots.*

To proof this we assume that $p(y) = \gcd(u(y), v(y))$ has a complex root $y = \omega + i \cdot \xi$. As $u(y)$ and $v(y)$ are real polynomials (i.e. its coefficients are real numbers) the polynomial $p(y)$ is also a real polynomial. Hence $\bar{y} = \omega - i \cdot \xi$ is also root of $p(y)$. Thus the complex number $\sigma + i \cdot y = \sigma - \xi + i \cdot \omega$ is a root of $f(z)$. Also $\sigma + i \cdot \bar{y} = \sigma + \xi + i \cdot \omega$ is root of $f(z)$. Since $f(z)$ is also a real polynomial the complex conjugate numbers $\sigma - \xi - i \cdot \omega$ and $\sigma + \xi - i \cdot \omega$ must also be roots of $f(z)$. As we always proceed to find the rightmost roots this can not happen. If we find σ of $f(z)$ there can be no root of $f(z)$ with real part $\sigma + |\xi|$. Q.E.D.

We now show the functioning of the algorithm with examples which explain the specific properties of the method.

Example 2 *Consider $f(z) = z^3 + 4z^2 + 5z + 6$. The Cauchy radius equals 7, i.e. $f(z+7)$ is a Hurwitz polynomial and $f(z-7)$ is not a Hurwitz polynomial. All roots lie within the radius $r_C = 7$ in the complex plane. $f(z)$ is tested and found to be a Hurwitz polynomial, so the real part of the rightmost zero lies in the interval $(-r_C, 0)$. After some further bisection iterations we find the real part $\sigma = -\frac{1}{2}$. Evaluating $f(-\frac{1}{2} + i\,y)$ we obtain*

$$f\left(-\frac{1}{2} + i\,y\right) = -\frac{5}{2}y^2 + \frac{35}{8} + i \cdot (-y^3 + \frac{7}{4}y)\ .$$

Hence $u(y) = -\frac{5}{2}(y^2 - \frac{7}{4})$ and $v(y) = -y \cdot (y^2 - \frac{7}{4})$ from which we get $gcd(u, v) = y^2 - \frac{7}{4}$. Thus we find $y = \pm\frac{\sqrt{7}}{2}$ and the two conjugate zeros are given by $-\frac{1}{2} \pm i\,\frac{\sqrt{7}}{2}$. After deflation the third zero -3 follows.

Example 3 *Consider* $f(z) = z^5 + 6z^4 + 21z^3 + 44z^2 + 56z + 40$. *The bisection algorithm leads to the real part* $\sigma = -1$. *Evaluating* $f(-1 + i\,y)$ *we obtain*

$$f\,(-1 + i\,y) = y^4 - 7y^2 + 12 + i \cdot (y^5 - 7y^3 + 12y)\ .$$

Hence $u(y) = y^4 - 7y^2 + 12$ *and* $v(y) = y^5 - 7y^3 + 12y$ *from which we get* $gcd(u, v) = y^4 - 7y^2 + 12$. *Thus we find* $y \in \{\pm 2, \pm\sqrt{3}\}$ *and the four conjugate complex zeros are given by* $-1 \pm i\,2$ *and* $-1 \pm i\,\sqrt{3}$. *After deflation the fifth zero* -2 *follows.*

The next two examples show that $u(y)$ or $v(y)$ can be identically equal to zero. This does not pose any problems, in fact it fits well into the computation of $gcd(u, v)$ as the greatest common divisor of u and v can be computed recursively using $gcd(u, v) = gcd(v, u \bmod v)$ which eventually ends with a polynomial $gcd(d(y), 0)$, where $d(y)$ is the greatest common divisor polynomial. Also note that $\gcd(w_1(y), w_2(y)) = \gcd(w_2(y), w_1(y))$.

Example 4 *Consider* $f(z) = z^5 + 5z^4 + 17z^3 + 31z^2 + 38z + 20$. *The bisection algorithm leads to the real part* $\sigma = -1$. *Evaluating* $f(-1 + i\,y)$ *we obtain*

$$f\,(-1 + i\,y) = i \cdot (y^5 - 7y^3 + 12y)\ .$$

Hence $u(y) = 0$ *and* $v(y) = y^5 - 7y^3 + 12y$. *As* $gcd(0, v) = v$ *we get* $gcd(0, v) = y^5 - 7y^3 + 12y$ *which delivers the same complex roots as the previous example plus an additional real zero at* -1 *from* $y = 0$.

Example 5 *Consider* $f(z) = z^6 + 6z^5 + 22z^4 + 48z^3 + 69z^2 + 58z + 20$. *The algorithm leads to the real part* $\sigma = -1$. *Evaluating* $f(-1 + i\,y)$ *gives*

$$f\,(-1 + i\,y) = -y^6 + 7y^4 - 12y^2\ .$$

Here $v(y) = 0$ *and* $u(y) = -y^6 + 7y^4 - 12y^2$. *Hence* $gcd(u, 0) = -y^6 + 7y^4 - 12y^2$. *The factor* y^2 *indicates that we have a double real root at* $\sigma = -1$. *The remaining equation* $-y^4 + 7y^2 - 12$ *leads to the same complex zeros as in the preceding two examples.*

Example 6 *Let* $f(z) = z^5 + z^4 + 4z + 4$. *The bisection algorithm delivers the real part* $\sigma = 1$ *of the rightmost zero. Then*

$$f(1 + i\,y) = 6y^4 - 16y^2 + 10 + i(y^5 - 14y^3 + 13y),$$

hence $u(y) = 6y^4 - 16y^2 + 10$ *and* $v(y) = y^5 - 14y^3 + 13y$ *which leads to* $gcd(u, v) = y^2 - 1$. *This delivers the two complex conjugate zeros* $1 \pm i$. *Deflation of these zeros gives the polynomial* $z^3 + 3z^2 + 4z + 2$. *The bisection algorithm then gives the real part* -1 *and* $(z^3 + 3z^2 + 4z + 2)|_{z=1+iy}$ *yields* $0 + i\,(-y^3 + y)$ *which leads to the zeros* -1, $-1 \pm i$.

3 Finite Precision Arithmetic

To complete the presentation we have to say a few words about the finite precision arithmetic which is used in actual numerical computations. We therefore give a pseudo-code for recursively computing the greatest common divisor of two polynomials using finite precision. Polynomials are assumed to be represented as arrays. The identity polynomial 1 is returned as [1].

Algorithm 2: GCD Computation with Finite Precision

```
gcd(a(z), b(z))
  if almostzero(a(z)): return b(z)
  if almostzero(b(z)): return a(z)
  if deg(a(z)) == 0 or deg(b(z))==0: return [1]
  return gcd(b(z), a(z) mod b(z))
```

The function almostzero($p(z)$) checks that all coefficients of the input polynomial $p(z)$ have absolute values less than a value ϵ_2. The error bound ϵ_2 (as the value of ϵ above) takes into account the kind of problem one has to solve. Usually ϵ_2 is set somewhat greater than ϵ. In a typical application in filter design for finding the poles of a transfer function (with simple complex roots and possibly one real root) it is sufficient to find the poles with 5 digits. So we can set $\epsilon \approx 10^{-7}$ and $\epsilon_2 \approx 0.5 \cdot 10^{-5}$ with double precision arithmetic. Of course for other problems the situation may be completely different. The value ϵ must not be set too small, in particular if no multiprecision arithmetic is used, as the Hurwitz test for too small values of ϵ can rapidly come to its numerical limit[3]. The selection of ϵ is discussed in the second to last section below.

For a real zero of $f(z)$ it is not necessary to perform the greatest common divisor computation $\gcd(u, v)$ since for such roots we have y as factor of $u(y)$. The polynomial $v(y)$ is an odd polynomial and is also divisible by y. For a real zero with multiplicity d we can easily check wether the first d coefficients of both $u(y)$ and $v(y)$ are smaller than ϵ_2. We show this finite precision by reconsidering example 5:

Example 7 *Let* $\epsilon = 10^{-7}$ *and* $\epsilon_2 = 0.5 \cdot 10^{-5}$ *with* $f(z) = z^6 + 6z^5 + 22z^4 + 48z^3 + 69z^2 + 58z + 20$. *The bisection algorithm for determining* σ *gives* $\sigma = -0.9999999729916453$. *Evaluating* $f(\sigma + i\,y)$ *with this value gives*

$$f\,(\sigma + i\,y) = u(y) + i\,v(y)\ ,$$

where

$$u(y) = -y^6 + 7.000000000000011 \cdot y^4 - 12.000000000000025 \cdot y^2 + 7.105427 \cdot 10^{-15}$$

and

$$v(y) = 1.620501 \cdot 10^{-07} \cdot y^5 - 7.562339 \cdot 10^{-07} \cdot y^3 + 6.482005 \cdot 10^{-7} \cdot y$$

Without entering the gcd computation, we get the double real root σ *as the coefficients at* y^0 *and* y^1 *of both polynomials* $u(y)$ *and* $v(y)$ *are much smaller than* $0.5 \cdot 10^{-5}$. *The distortion of the coefficients of* $(u(y) - 7.105427 \cdot 10^{-15})/y^2$ *is negligible and thus we find the correct imaginary parts of the remaining 4 complex zeros by solving a quadratic in* y^2.

Before we present the final algorithm for computing all roots of a polynomial, we present an algorithm which finds all roots of $p(y) = \gcd(u(y), v(y))$, i.e. an algorithm which finds all roots if we know that the input polynomial has only real roots. The algorithm is straightforward. The roots are collected in the array *zeros* which is initially set to the empty array []. The zeros found are appended with the command zeros.append(σ).

[3]As a rule of thumb we recommend to set ϵ in the order of $r_C \cdot 10^{-\frac{L}{2}}$ if L digits arithmetic is used. If a higher precision is desired and increasing L no option one may still use the method and add some root polishing using Newton's or Bairstow's method.

The polynomials $u(y)$ and $v(y)$ are also stored in arrays. The lowest coefficients of both polynomials must be zero at least once for a single root and are accordingly deleted with the commands del u[0] and del v[0]. The last line recursively calls the deflated input polynomial, where $\sqcup$ denotes array concatenation.

Algorithm 3: Finding all roots of a polynomial which has real roots only

realroots($f(z)$)
  $n \leftarrow$ deg(($f(z)$)
  zeros=[]
  if $n < 1$: return zeros
  if $n == 1$: return [-f[0]/f[1]]
  σ =bisect($f(z)$)
  $g(z) \leftarrow f(z + \sigma)$
  $(u(y), v(y)) \leftarrow (\Re\{g(iy)\}, \Im\{g(iy)\})$
  while abs($u[0]$) $< \epsilon_2$ and abs($v[0]$) $< \epsilon_2$:
    zeros.append(σ)
    $f(z) \leftarrow f(z)/(z - \sigma)$
    del u[0]
    del v[0]
  return zeros $\sqcup$ realroots($f(z)$)

We are ready to give a complete algorithm for finding all roots of a polynomial $f(z)$. In addition to the previous algorithm the imaginary parts are processed. The conjugate roots found are deflated by dividing through the quadratic equation which contains both conjugate roots (i.e. we are using real arithmetics only). The last loop checks whether the solution(s) y are greater than ϵ_2. Thus ϵ_2 controls the precision of the imaginary parts and ϵ the precision of the real parts.

Algorithm 4

```
roots(f(z))
  n ← deg((f(z))
  zeros=[]
  if n < 1: return zeros
  if n == 1: return [-f[0]/f[1]]
  σ =bisect(f(z))
  g(z) ← f(z + σ)
  (u(y), v(y)) ← (ℜ{g(iy)}, ℑ{g(iy)})
  d ← 0
  while abs(u[0]) < ε2 and abs(v[0]) < ε2:
    zeros.append(σ)
    f(z) ← f(z)/(z − σ)
    del u[0]
    del v[0]
    d ← d + 1
  p(y) ←gcd(u(y), v(y))
  if deg(p(y)) ≥ 1:
    Y ← realroots(p(y))
    for y ∈ Y:
      zeros.append(σ + i · y)
      if y > ε2:
        f(z) ← f(z)/(z^2 − 2σz + σ^2 + y^2)
        d ← d + 2
  if d < n: return zeros ⊔ roots(f(z))
  return zeros
```

We now prove that our algorithms find all roots with certainty.

Theorem 2 *If the number of digits used and the values ϵ and ϵ_2 are selected adequately, then the above algorithms always find all roots of the n-th degree polynomial $f(z)$.*

Proof: Any n-th degree polynomial $f(z)$ has exactly n roots within the Cauchy radius r_C. If the number of digits is properly chosen, Hurwitz's test always tests correctly whether all roots of the polynomial $f(z+\sigma)$ are in the left half of the complex plane or not. Hence the bisection algorithm 1 always finds the real part σ of the right-most zero of $f(z)$ to any precision required since for a given value ϵ the size of the interval in which σ lies is halved with each iteration. From $f(\sigma + z)$ we obtain the two polynomials $u(v)$ and $v(y)$. If the coefficients are known with sufficient precision, we can find the greatest common divisor of $u(y)$ and $v(y)$ from which the imaginary part(s) of the zero(s) can be found. *Q.E.D.*

In practice one has to be very careful about ill-conditioned polynomials. This is a very essential problem for any root finding algorithm. Wilkinson [7] has some interesting examples where very small errors in the coefficients can change some real roots of a polynomial to complex roots. Nevertheless, this is no principal hurdle for the above algorithms. By setting the values ϵ and ϵ_2 appropriately and/or using multiprecision arithmetic these problems of ill-conditioned polynomials disappear.

If one does not have much information about the polynomials whose roots are to be found, the parameter ϵ poses a problem if a fixed number L of digits is used and can not

– for whatever reason – be increased. Namely the Hurwitz test can fail, if ϵ is chosen too small and Algorithm 1 may run into an infinite loop. Therefore it is best to limit the number of iterations n_{itmax} as in the following Algorithm 1a. This algorithm for a fixed number of digits in the worst case does not reach the error bound ϵ but at least finds a good approximation for σ. Then it is also straight-forward to adapt the value ϵ_2 to get quite good initial values for the roots which eventually can be found using classical root-polishing algorithms. A square root bound for the selection of ϵ is given in the next section.

Algorithm 1a: ϵ-tolerant Bisection Algorithm for Determination of Real Part σ

bisect($f(z)$)
  $r_C \leftarrow$ cauchyradius($f(z)$)
  $(a, b) \leftarrow (-r_C, r_C)$
  for $k \leftarrow 1$ to n_{itmax}:
    $\sigma \leftarrow (a + b)/2$
    if $|b - a| < \epsilon$: return σ
    $g(z) \leftarrow f(z + \sigma)$
    if hurwitz($g(z)$): $b \leftarrow \sigma$
    else : $a \leftarrow \sigma$
  return $\frac{a+b}{2}$

4 A square root bound for ϵ

We give a recommendation as to the selection of ϵ under the assumption that we use L-digit precision[4]. We recommend setting

$$\epsilon \approx r_C \cdot 10^{-\frac{L}{2}} \ .$$

To justify this selection we assume that $f(z)$ contains two real roots which are very close together say at $c+\epsilon_0$ and $c-\epsilon_0$. Then $f(z)$ has the quadratic $(x-(c+\epsilon_0))\cdot(x-(c+\epsilon_0)) = x^2 - 2c\,x + c^2 - \epsilon_0^2$ as factor. If $\epsilon_0 < c \cdot 10^{-L/2}$ and c has L digits mantissa, then we have for the term $c^2 - \epsilon_0^2 < c^2(1 - 10^{-L})$, which means that ϵ_0^2 does not appear in the L-digit representation of the number $c^2 - \epsilon_0^2$. It also does not appear in the number $c^2 + \epsilon_0^2$. This means that it is impossible to distinguish for example the values $c \pm \epsilon_0$, or $c \pm i\,\epsilon_0$ or a double root at c. Hence the precision is bounded from below by the square root bound $c \cdot 10^{-L/2}$. The recommendation above then follows by replacing c by the Cauchy radius r_C. The value of ϵ_2 is not particularly critical, it should be set somewhat greater than ϵ (say $\epsilon \approx 10^l \cdot \epsilon$ with l a small integer).

This recommendation will do for almost all polynomials. If one expects particular nasty ill-conditioned polynomials (for example clusters of roots which are very close together) one may further increase the value of ϵ (say to $\epsilon \approx r_C \cdot 10^{-L/3}$). Alternatively, of course, one may increase the value of L.

[4]The common double precision data format uses 53 bits for the mantissa which correspond to roughly 16 digits ($L = \log_{10} 2^{53} \approx 15,95$).

5 Conclusion

An algorithm for the solution of polynomial equations has been presented which finds real and complex roots with certainty. It uses a bisection algorithm to find the real part of the rightmost root. The imaginary part is found using gcd-computations.

Another advantage of the algorithm is that all roots of a polynomial $f(z)$ including the complex roots can be found by using essentially only real arithmetics. Even determining $f(\sigma + i\,y)$ does not need complex arithmetic as we can compute $g(z) = f(\sigma + z)$ and later we can separate the polynomials $u(y)$ and $v(y)$ by essentially reading off the corresponding coefficients from the polynomial $g(i\,y)$ which can be done without complex arithmetic.

If L-digit arithmetic is used the algorithm typically delivers $L/2$ correct digits. If higher precision is needed one has to increase L i.e. use multiprecision arithmetic which nowadays is readily available with public domain software.

References

[1] F.S.Acton, "Numerical Methods that work", The Mathematical Association of America, Washington D.C., revised edition 1990.

[2] K.Huber, "All-Pole Filters Matched to Specifications", Journal of the Audio Engineering Society, Vol. 61, No. 12, pp. 1022-1025, December 2013.

[3] A.Hurwitz, "Ueber die Bedingungen, unter welchen eine Gleichung nur Wurzeln mit negativen reellen Theilen besitzt." Mathematische Annalen, Band 46, 1895, pp. 273-284.

[4] W.B.Jones, W.J.Thron, "Continued Fractions", Cambridge University Press, 1980.

[5] E.Laguerre, "Sur une methode pour obtenir par approximation les racines d'une equation algebrique qui a toutes ses racines reelles", Novelles Annales de Mathematiques, 2-eme serie, t. XIX, 1980, dans Oeuvres de Laguerre, Tome 1, second edition Paris, 1898, reprint Chelsea Publishing Company Bronx, New York,1972, pp. 87-103.

[6] D.H.Lehmer, "A Machine Method for Solving Polynomial Equations", JACM, Vol. 8, 1961, pp. 151-162.

[7] J.H.Wilkinson, "Rounding Errors in Algebraic Processes", HMS Office London, 1963.

[8] E.Zeheb, "On the Largest Module of Polynomial Zeros", IEEE Transactions on Circuits and Systems, Vol.38, No. 3, March 1991, pp. 333-337.